VASAVI COLLEGE OF ENGINEERING (Autonomous)
IBRAHIMBAGH, HYDERABAD – 500 031
Department of Computer Science & Engineering

INNOVATION IN TEACHING

Course Name: COMPUTER ARCHITECTURE **Faculty Name:** Mamatha K

Topic Name: Machine Instructions **Year/Sem:** III-Sem

Innovative Teaching Aid/Tool Used: LMC-Simulator

Description of the Tool:

The **Little Man Computer** (**LMC**) is an instructional model of a computer, created by Dr. Stuart Madnick in 1965.The LMC is generally used to teach students, because it models a simple von Neumann architecture compute. Which has all of the basic features of a modern computer. It can be programmed in machine code (albeit in decimal rather than binary) or assembly code.

Little Man Computer - LMC - is a simulator that mimics the modern computer architecture, known as **von Neumann architecture.** It was a brainchild of Dr Stuart Madnick, invented in 1965; Since it can model the modern computer, it is still widely used as a teaching tool.

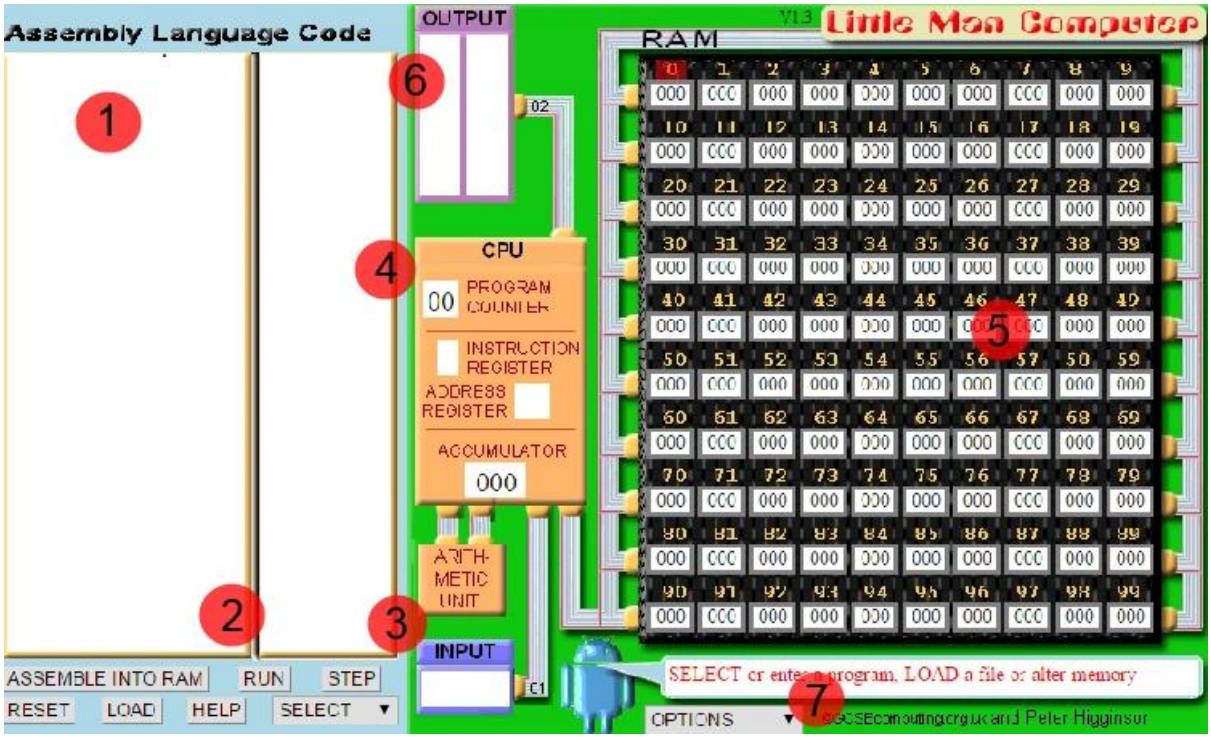
Von Neumann Architecture

The Von Neumann architecture, illustrated in the following image, consists of five major components with machine equivalent of division of labour. They are,

1. Control Unit
2. Logic Unit
3. Memory
4. BUS
5. Input-Output Unit

Input-Output Unit
The unit allows a user interact with the computer while providing it with an input in anticipation of an output.
Control Unit
This unit deals with the handling of instructions, processing of the data, storing it in memory and reading it back from it at the right moment.
Logic Unit
This unit is responsible for carrying out basic mathematical operation such as addition, subtraction etc.
Memory
This section stores data and the instructions to deal with data.
BUS
This is the machine equivalent of umbilical chord - the connector of the other four parts to the motherboard.

The LMC simulator takes the following form:



First of all, type in the following code in the **LMC** as shown in the animation in order to see how it works:

INP
STA 20
OUT
HLT

Tool Usage in Teaching:

The LMC is based on the idea of'Little Man' acting like the control unit of a CPU, fetching instructions from RAM, decoding and executing them as well as managing the input and output mechanisms.

Understanding the LMC simulator

- The 100 memory **addresses** in the **computer memory** are numbered 0 to 99 and can each contain a '**machine code** 'instruction or data.
- Each **assembly language instruction** is made up of a 3-letter mnemonic (which represents the operation code), usually followed by the memory address of the data the CPU is to act on (this is called absolute memory addressing).
- Pressing the **Assemble Program** button translates the **assembly language instructions** into '**machine code** 'and loads them into **RAM**. it also resets the Program Counter to zero.
- The **Input box** allows the user to enter numerical data (-999 to 999) while a program is running and load it into the **accumulator**.
- The **Output box** can output the contents of the accumulator while a program is running.
- A **RAM memory address** that is used to store data can be given a meaningful label. Data can also be stored in these named address locations.

- The results of any **ADD** or **SUBTRACT** instructions are stored in the accumulator.
- The **Program Counter** stores the memory address of the instruction being carried out. It will automatically increment by 1 after each instruction is completed.
- If the CPU receives a **non-sequential** instruction to **branch** (BRP, BRP or BRZ) then the Program Counter is set to the memory address of that instruction.
- Branch instructions are set to branch to a labelled memory location.
- To **restart** a program, the Program Counter is reset to 0.

When assembled, each **assembly code instruction** is converted into a 3 digit 'machine code' instruction (1 digit for the instruction and 2 for the memory address). The 3 digit 'machine code' instructions are then loaded into RAM, starting at memory address 0.
Any data is also loaded into memory **at the memory address corresponding to the location of the data in the program** (i.e. if the 5th line of the assembly language program was data then this data would be loaded into address 4, because memory address start at 0 not 1)
The 'Little Man' can then begin execution, starting with the instruction in RAM at memory address 0.
The 'Little Man' performs the following steps to execute a program:

1. Check the **Program Counter** so it knows the memory address to look at.
2. **Fetch** the **instruction** from the memory address that matches the program counter.
3. **Increment** the Program Counter (so that it contains the memory address of the next instruction).
4. **Decode** the instruction (includes finding the memory address for any data it refers to).
5. If required by the instruction code, **fetch the data** from the memory address found in the previous step.
6. **Execute** the instruction and if necessary, set the **Program Counter** to match any **branch** instructions.

Reference(s):

LMC Instruction set:
http://www.yorku.ca/sychen/research/LMC/LMCInstructions.html
LMC Simulator: http://www.yorku.ca/sychen/research/L...
http://peterhigginson.co.uk/LMC/

Video Link: https://www.vivaxsolutions.com/images/lmc-2.webm